05/18/12 Justin's To Do List

MAKE JUSTIN'S PRICELIST - THAT'S WHAT HE REALLY WOULD LIKE

see where timers code is, add explanations about how to really use the CTC modes.

make this more useful for Justin.

pin 21 is GND, check Justin's wiring

05/20/12

new program name - AnyNote64

list of changes to implement in the mini_speaker program:

PROGMEM - keep lookup tables in flash memory
    make the arrays as const in flash - PROGMEM

64 sample Sine (or Cosine) Wave lookup table
    0-255 levels
        maybe reduce the Sine Wave Lookup and PlayArray to 64 entries (max).
        what would be the difference in quality?
            6 bit THD is 25dB.
            http://www.avrfreaks.net/index.php
                ?name=PNphpBB2&file=printview&t=40426&start=0
    IF I MAKE THE 64 SAMPLE COSINE WAVE VERSION.
        FOR THE HIGH NOTES, I COULD ADD A 32 SAMPLE AND A 16 SAMPLE VERSION FOR
            QUICKLY MAKING THE MULTI-CYCLE SvPlayArrays.
        (or 64 sample two sine wave, and 64 sample 4 sine wave)
    each cycle should have the same area under the curve RMS
    64 bit Cosine lookup table
        64 bit - two cosine lookup table, 4 cosine
    each cycle should have the same area under the curve RMS
    see what happens with the Cosine versions on graphing spreadsheet

use reciprocal table and multiply in place of the division of
    Counter/TotalElements
    RECIPROCAL 256 lookup for using multiply instead of divide
        put this into graphing spreadsheet, see what happens with different
            lengths of PlayArrays, ArrayElements. compare two version of the
            calculation.
        use the values from 128-255
        when calculating ratios of the counter to the number of items, the
            result will be a 16 bit number
        two 16 bit values, multiply into a 16 bit result.
        for the mini_speaker project.
        for calculation the sine wave lookup table into another length Sine
            Wave.

by octave the length of the new sine waves are from 128 sample to 256 sample (when using the 256 sample sine wave lookup table)

for the 64 sample sine wave lookup table, the lengths of the new Sine Waves are 32-64.

(SvArrayElements[SvNextLiveArr] + 1)

only need the reciprocals of 129-256

use the reciprocal result left-shifted 16 bits (* 65536)

recip 128 = 512, 129 = 508, 255 = 257, 256 = 256

127 * 512 = 65024, 128 * 508 = 65024, 254 * 257 = 65278, 255 * 256 = 65280

reduce recips by 2 (>> 1) ?

maybe this could make the 9 bit by 8 bit multiply just an 8 x 8 but it would probably ruin the needed accuracy, see what happens

STOP AT NOTE 119. 120 notes total. 12 octaves.

WRITE A VERSION THAT PLAYS ANY TONE.

have main determins SvHighNOTLow, SvNoteFactor and ArrayElements.

send those values and SvNoteDuration and SvNoteVolume to new function.

SfToneGeneratorAny.

use a struct:

SvToneValues

SvHighNOTLow

SvNoteFactor

ArrayElements

SvNoteDuration

SvNoteVolume

ANY TONE - struct

main function decodes the NoteNum

combine all of the Volatile Arrays[SvLiveArr/SvNextLiveArr] into a Struct

linked list of Structs, use pointers

make transition effect between notes (fade and/or phase matching)

try a slope between ending of note and start of next note

figure out to make transition happen in time the current note to end

is there a lowest slope that doesn't sound like a click (or anything else) that I could use to transition between tones?

what does other music software do between notes?

```
      probably doesn't just "click" over too the next note


GET PHASE MATCHING TO WORK.
      I want to get this to happen just for the exercise of doing it
save ending phase, to match next note's starting point in phase
      at end of note, save the phase of the last step (sample).
            start next step (sample) at the same place in the phase.
page 7 of 7 markup in ISR
      SvNextPWMValue = SvPlayArray[SvNextLiveArr][ ** ];
            ** change with phase and calculate new SvArrayLiveCounter to
                  point to correct earliest phase in NextLiveArr
            ** next phase corrected SvLiveCounter value
      determine phase at end for low Notes, it's the SvArrayLiveCounter value
            divided into the number of steps
it would be quicker to have matching arrays to PlayArray[][] that contain
       the current phase
      (this is the first (1 / (1 << NoteFactor) ), for high notes
            PhaseNum / (1 << NoteFactor) )
Phase = (CurrentStepCount << 8) / TotalSteps in current SvPlayArray
      NewPhase (starting position) =
            (Phase * TotalSteps in new PlayArray) >> 8
      FOR 256 SAMPLE SINE WAVE
Phase matching
      increased phase accuracy
      6 bit "push" - get 4 bit highest note phase value
for highest notes with 4 sample cycles,
      64 sample sine/cosine table will allow my calculation to get a 4 bit
            phase value, it was only 2 bit.
      use this to get PlayArray values accurately (much better anti-aliasing)
      with 64 sample Sine/Cosine lookup table, the 6 bit counter can "push"
            up to 8 bit to get a more accurate phase position
            for 4 sample cycle, it can get 16 position level
only use 4 bit phase info?
      maybe there is just some way I can save the level and move to an
            appropriate level in the next PlayArray
            Phase position won't be useful for level matching with complex
                  waveforms anyway
the phase matching won't work for different volumes
```

hookup Arduino communication, use for Diagnostics
- show interesting values of variables
- AVR - Arduino - PC (log data)
  - PC (control file) - Arduino - AVR
- make the communication to Arduino, see what happens with 16 bit multiply.
  - 8 bit multiply. 8 bit result?
  - try to speed up my 8 and 9 bit multiplies.
  - assembly routine, AVR has 8 bit x 8bit to 16 bit
- check if all places that I want 16 bit result really give 16 bit result
- AVR - ARDUINO - COMPUTER I/F
- AVR - Arduino - computer
- AVR monitor - status

add notes, different volumes (subtract)
- when combining waveforms, high notes can just be added together,
  - 78K step by 78K step, Low Notes need to be divided out by 2^NoteFactor.
- it won't be possible to get a 129-256 step repeating wave,
  - it would have to be calculated for each 78K step.
- if two waves, add their PlayArray values and right shift 1.
  - for three waves, uh...
    - I think you just have to divide by three.
  - etc.
- Two tones (3 tones, 4, 8... ?)
  - how much can be done with this chip?
- for the multi-note combining have the combining function make 256 step
  - ActualPlayArrays that play a sample per 78K PWM cycle (step)

make actual PlayArrays - short ISR
- probably a good idea to do this anyway
- a very short ISR will let the rest of the program run more longer and the
  - context switching wont be such a problem
- can I make a version that plays 256 sample ActualPlayArrays in the ISR?
  - 305 sample?
- need to make sure that I always have an ActualPlayArray ready to go.
- maybe get a set of APA's ready?
- sequentially order APA's with pointers?
  - linked list
- need a new 305 samples every 78,080 clock cycles
  - 3.904 ms

- does a shorter ISR have a shorter context switch in the assembly?
- my sp_timers RTC has an interrupt every 10uS,
  - I can insert a part that runs often enough and just looks at what needs to be done and sets flags.
  - if I'm coming around my main loop fast enough, I can do the flag things there anyway.
  - just keep ALL things to do broken into small enough chunks
- I think that the largest reciprocal numbers that I need are 9 bit, if so,
  - put the lower 8 bits into my lookup table for multiplying since I will be right-shifting this result,
  - I would just need to add the "count" number to this result for when bit 9 is a 1.
- I really only need and 8 bit by 8 bit multiply into only the highest byte,
  - and an add if bit 9 is a 1
  - I could make this happen in assembly or a C library function thing
  - result is always 8 bits or less
- try to keep the ISR very short
- read through how long the ISR assembly is
  - see if making the ISR shorter, with less variables and other stuff,
    - makes the context switch shoter
    - less push and pop from stack

other waveforms than sine wave
- triangle, square, trapezoidal, etc.

emvelopes for any pattern changes
- frequency (logarithmic)
- volume (logarithmic)
- various envelope shapes parameters
- ADSR (attack, decay, sustain, release)
- various envelopes:
  - frequency, volume, waveform parameters
- ADSR (Brian)
  - audio parameters

envelope patching:
- wave output to envelope or other parameter modification

exponential lookup table
look into using exponential lookup table - volume
log - (adder snakes joke)

check if the SvNoteWaveLength78K[] values should be adjusted.
change the program to use only 12 values.

check if SvSineWaveLookup256 numbers should be adjusted.
the Sine Wave numbers should probably be adjusted, +/- 127 from 127 (or 128).
TRUNC((128 * SIN((2 * PI) * (step / TotalSteps))) + 127.5)
should I be using Cosine?
start at a high value.
what difference for low-sample quantity/high-frequency tones?

put (1 << SvNoteFactor[SvLiveArr]) into a variable to keep from having to recalculate it each time

markups on mini_speaker_gl printout Tue 15 Mar 2012 07:53 AM
page 5 of 7
shows what to use from SfToneGenerator to get SvPlayPhaseArray values
Phase of current sample step is saved in matching array, for use in ISR, to start next note in phase.
also mentions adding a temporary variable to eliminate a repeated calculation.
page 6 of 7
put (1 << SvNoteFactor[SvLiveArr]) -1) into another variable SvNoteFactorTime[] to keep from having to calculate it here in the ISR

SEE IF I'M GETTING THE CORRECT TONE. PLAY MIDDLE C.

read through the Makefile

fix descriptions in the SfToneGenerator function, they have old description of high-note and low-note amplifier circuit notes:

was thinking about putting my speaker "GND" at 2.5V. need to make circuit adjust
    for correct current/power

op amp circuit for 10KHz RC filter
- the transistor amp (better transistor amp)
- find real headphone amp circuits
    - 8 or 32 ohm

run audio circuit from analog power/gnd rails

for amplifier circuit, try to get best values to use for low-pass filter
- 20Hz-20KHz, what time constant should I use from the 78Khz PWM signal?
- other improvements to amp circuit
    - correct linearity
    - +/- voltage for headphones?

find another circuit simulator
- faster than the java sim
- use the real sim?
- SPICE
- ngspice - simulator
    - gspiceui
    - easyspice

additional notes:

BASICALLY, no result will contain more bits than the largest operand

LOOP COUNTING METHOD:

```
do the thing
compare to max (top) count
      if max (top) count, set to 0
            do something more?
            set flag?
      else increment
end
```

buy a couple more ATmega328s (P or not P ?)

gcc, avr-libc updates

bus pirate correct firmware?

use ADC port to read in values at summation/filter node, and get curve to adjust
    to flat scaled output values

find out how old 8 bit audio worked, what capabilities
    old video games

really learn transistos BJT/FET math

HACKMEM

simulavr, dbg

maybe I was wrong about the 8 bit multiply giving 8 bit result.
    in C ...
    page 4 of Essential C says (k * 1024) will be 16 bit if "k" is 16 bit

interval ratios - read that link

no multi-bit shift in AVR assembly?

I thought my shift commands would be really speedy, maybe they are not

remember to check the things that I keep messing up on

if statments use ==, not =

check variable sizes, uint8_t uint16_t

count from 0, count from 1

go through all of this again, and markup the width of the operands

variable sizes, uint8_t, uint16_t

read Essential C, about heap and stack

heap - variables

stack - status and intermediate terms in equations

I think that this is wrong from looking at the .lst files from the compiler

find instructions for high-performance microcontroller programming

is there a "size of array" that is easy to use?

check compiler files, verify that the variables and data are setup correctly

google - gcc.info standard names umulm3_highpart

umulm3_highpart - multiply, keep only the high part

get the saved bookmarks that I have made at Williamsons

review my bookmarks

qtiplot

ooh, I could just map 1/4 of the voltage levels on the Sine Wave.

1/4 a phase, add or subtract from 0, mirrored.

only need the sine wave values from 1 quadrant.

future changes and additions:

LCD DISPLAY

LCD I/F
- interface keyboard?

add multi-tone combining
- phase shift
- tone frequency shift
  - phase
  - combining note ratios
  - combining voice ratios
- add/subtract voices

voice playback (frequency match in time slices)
- vocoder-type effect?

play notes from keyboard
- MIDI keyboard?

map linear to log scale
- other scale mapping (exponential)
  - (same thing?)

limiters

low-pass, high-pass, middle

various multi-spectral

frequency shift

put my graphing spreadsheet calculations through an FFT, see if I'm getting these harmonics.

put a pure wave through and back with a good FFT analysis see what this looks like for a highest-notes.
(I think that FFT doesn't understand the phase of the cycles)

how pure a wave can be done at various frequencies using the 78,125Hz sampling?

and changing the length of the PlayArray

(Sine256 lengths are 129-256, Sine64 lengths are 33-64)

(look at that Nyquist Limit demo software, see if suggests anything.

effects with FFT

learn about 8 bit FFT

LCD display waveform